SPICAN

CANopen I/O-system (for analog inputs)

(applies to SPICAN, CRYSTAL-CAN and CRYSTAL-CAN-2 modules with SPICAN CANopen firmware)

H. Boterenbrood
NIKHEF, Amsterdam
January 2000

USER DOCUMENTATION Version 2.1

ABSTRACT:
This document describes the **SPICAN** system of controller and signal-conditioning module(s) (e.g **T-SENSOR**) in combination with ***CANopen*** application firmware for monitoring up to 192 analog input channels.
The **CRYSTAL-CAN** hardware is electronically almost identical (**CRYSTAL-CAN-2** is electronically fully identical) to SPICAN and the same *CANopen* firmware (with some limitations in case of **CRYSTAL-CAN**) is available for these systems.

Contents

TECHNICAL SPECIFICATIONS **2**

1 INTRODUCTION **3**

2 OPERATION **5**

2.1 INITIALISATION 5
2.2 READING ANALOG INPUT CHANNELS 5
2.3 OVER-LIMIT NOTIFICATION 7
2.4 SETTING UPPER LIMITS 8
2.5 STORING PARAMETERS 10
2.6 ADC RESET AND CALIBRATION 11
2.7 ADC CHANNEL NUMBERING SCHEME 11
2.8 EMERGENCY OBJECTS 12

3 OBJECT DICTIONARY **13**

REFERENCES **20**

APPENDIX A **LEDS, SWITCHES AND JUMPERS** **21**

APPENDIX B **CONNECTOR LAYOUT** **22**

Technical Specifications

SPICAN controller module:

- microcontroller: Philips P80C592 8-bit @ 16 MHz
- in-system-programmable via RS232 port
  (available user memory: 63.5 kByte flash-ROM, 48 kByte RAM)
- power requirement: ca. 200 mA @ +5V
- firmware versions *SPICAN v4.0* and up:
  CANopen device profile according to CiA DSP-401, CAN node-id between 1 and 127, CAN baudrate 125 or 250 kbit/s, minimum boot-up, default *CANopen* COB-ID distribution, 192 analog inputs mapped to up to 6, 12 or 24 ADCs, over-limit interrupt on first 64 channels (can be extended to all 192 channels), control and configuration of individual ADCs, non-volatile storage of parameters and settings

T-SENSOR module:

- 30 NTC-sensor inputs
- operating range: 0°-100°C
- power requirement: ca. 1.5 mA @ +5V
- conversion table ADC-count → temperature provided (in ASCII-format)
- accuracy: 0.3°C (0.1°C calibration precision + 0.2°C spread in NTC sensor (DC95-F-503-W, 5kΩ)
- resolution: ranging from ca. 1m°C at 5°C to 25m°C at 100°C
- drift: 1m°C/°C when regularly applying calibration (e.g. once per hour), otherwise 5m°C/°C

1 Introduction

From a hardware point-of-view the **SPICAN** system is a modular CAN-node consisting of a controller card and one or more I/O-cards or -modules which are controlled and read out via a serial connection (SPI or MicroWire type).

The controller card (Eurocard format) contains the microcontroller and CAN-interface. It is built around a 16-MHz Philips **80C592** 8-bit microcontroller with on-chip CAN controller. It provides 48 kByte of user program memory and 63.5 kByte of user RAM [1]. Program code (in standard Intel Hex format) can be downloaded directly via the RS232 port.

A **T-SENSOR** I/O-card (Eurocard format) has been developed, suitable for connecting 30 NTC-sensors. The T-SENSOR card is built around the 16-bit **CS5525** ADC [2], which is controlled through a 3-wire serial interface (SPI).

The SPICAN controller card has been designed specifically to control any number of **SPI**-controlled I/O modules that it can select individually using an 8-bit select-port, as shown in Figure 1. SPI (*Serial Peripheral Interconnect*) is a simple serial point-to-point connection between devices. There are many types of chips with SPI-like interfaces available on the market.

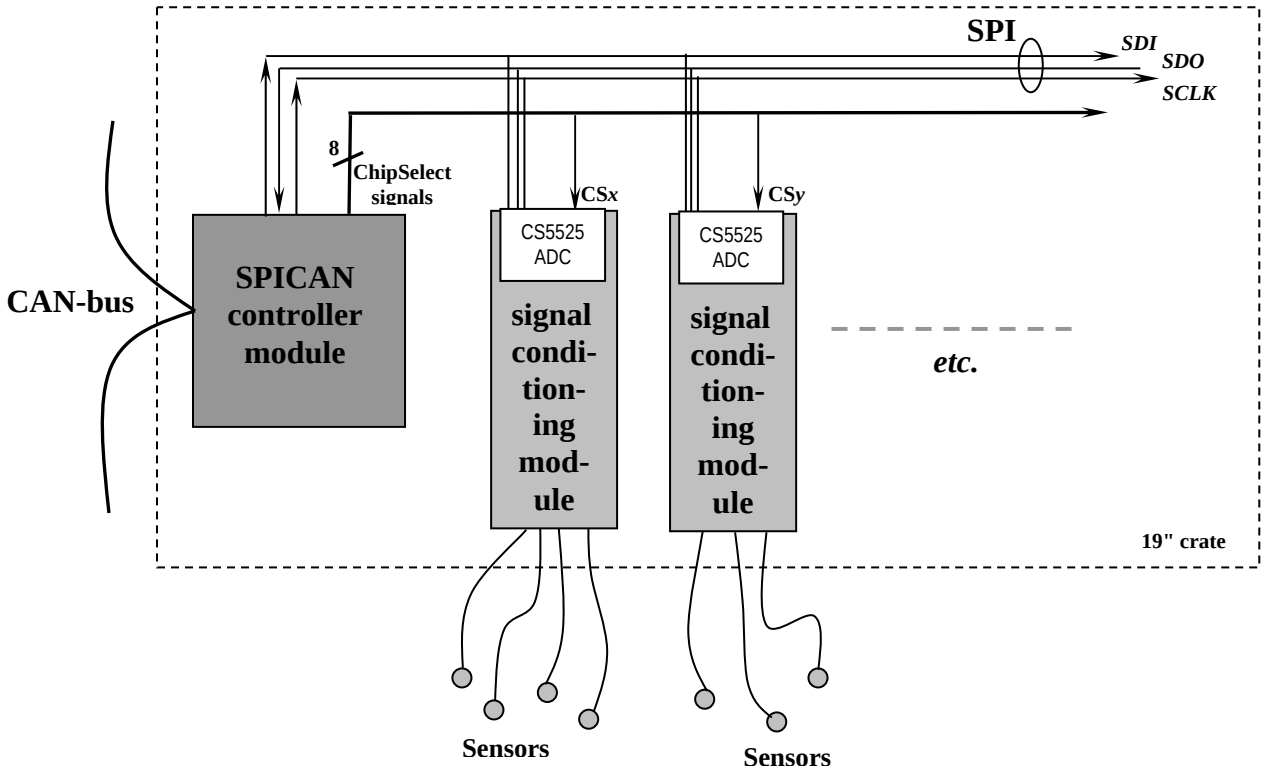


Figure 1. SPICAN system.

Apart from the signals used to control a number of external SPI-devices several other signals provided by the on-board microcontroller are available on the SPICAN controller card connector, e.g. eight 10-bit ADC inputs (connected to the 80C592 on-chip ADC). These signals can be used by applications if appropriate application software is written, using a development toolset for 8051 microcontrollers.

The hardware differences between the **SPICAN** system and the **CRYSTAL-CAN** module with its matching T-sensor and B-sensor conditioning modules are:

- the mechanical format
  - A CRYSTAL-CAN node consists of individual boxes: one controller box and one or more signal-conditioning boxes, connected by flatcable.
    A SPICAN node consists of one controller card (single Eurocard format) and one or more signal-conditioning cards sitting in a 19' crate, connected via a backplane.
  - A CRYSTAL-CAN node must get its power via the CAN cable.
    A SPICAN node has a power-supply in its crate or optionally can be powered via the CAN cable.
- the SPICAN controller card contains an EEPROM for storing settings and configuration data; the CRYSTAL-CAN is not equipped with an EEPROM so that settings/configurations cannot be stored; if settings different from the power-on default are required on the CRYSTAL-CAN node they have to be reconfigured at every power-up/reset of the node (or the controller has to be reprogrammed to use appropriate defaults).

Of course it is the software that determines the functionality of this device and this document describes the application firmware that has been developed for a SPICAN system with multiple analog inputs from multiple CS5525-ADC-based I/O-cards. It can monitor up to 192 analog input channels connected to up to 24 Crystal CS5525 ADCs. It features a mechanism whereby up to 64 analog inputs are checked against a per-channel-configurable upper limit. A CAN-message is generated if a limit is crossed (from under- to overlimit as well as from over- to underlimit).

This application is currently in use for monitoring temperature sensors, B-field sensors and voltages and currents; only the signal-conditioning hardware is different in each of these applications.

Monitoring and other communication takes place via the CAN-bus, using the standardized high-level CAN-bus protocol **CANopen** ([3], [4], [5]). For a concise description and overview of the *CANopen* protocol see [6].

2 Operation

The following sections show examples of the (*CANopen*) CAN-messages required to control and operate a SPICAN (or CRYSTAL-CAN) system.
In the examples below the following assumptions are made:

- CAN Node-ID of the controller is 5.
- two ADCs are connected (i.e. two T-SENSOR modules)
- each ADC serves 32 input channels (the last two of which are the 100°C and 0°C calibration inputs), but the number of used ADC channels has been set to 30 (thus skipping read-out of the calibration inputs).

2.1 Initialisation

After power-up, watchdog reset, manual reset or *CANopen* initiated reset actions the SPICAN node sends a so-called *Bootup* message (defined by the *CANopen* standard) as soon as it has finished its initialization; this is a CAN-message with the following syntax:

SPICAN (NMT-Slave) → Host (NMT-Master)

COB-ID	Byte 0
0x700 + *Node_ID*	0

In case of a watchdog or manual reset the Bootup message is followed by a *CANopen* Emergency message, as listed in the table in section 2.8.

2.2 Reading Analog Input Channels

Before any input channels can be read the connected *CANopen*-nodes have to be set into *Operational* state using the following 2-databyte ***NMT*** message:

Host (NMT-Master) → SPICAN (NMT-Slave)

COB-ID	Byte 0	Byte 1
0x000	1 (*Start_Remote_Node*)	5 (*Node-ID*) or 0 (all nodes)

There is no reply to this message.

The analog inputs are read out using the *CANopen* **PDO** mechanism. A PDO message is a non-confirmed CAN-message with one sender and one or more receivers, containing no protocol overhead, only data (1 to 8 bytes). It is assumed that receivers of a PDO message know the meaning of the data content of a PDO message.

The SPICAN application uses a PDO containing 4 bytes for every analog input. The CAN-identifier used for this PDO is the socalled *2nd-transmit-PDO* of the CANopen Predefined Connection Set, which is the default PDO used for analog inputs according to *CANopen* (meaning COB-ID = 0x280 + Node_ID).

Node 5 will produce the following 4-databyte PDO:

SPICAN (node #5) → Host

COB-ID	Byte 0	Byte 1	Byte 2-3
0x285	Channel Number	Channel status	ADC value

with:

Channel Number: runs from 1 to 30 and from 33 to 62 (skipping the calibration inputs at channel 31+32 and at 63+64, if the node is configured as described earlier)

Channel status: **0xf0**: OKAY; **0xf1, 0xf2, 0xf3, 0xff**: ERROR.

ADC value: 16-bits value, LSB in byte 2, MSB in byte 3 (ADC value 0x8000 in combination with channel status 0xf1 means no sensor is connected to the particular input)

The way in which all 60 analog inputs of the example CAN-node can now be read out depends on the *transmission-type* of the PDO. The user reads out the analog inputs according to the PDO transmission-type the node has after power-up. Alternatively the user can set the transmission-type to the required value by writing to the node's Object Dictionary (OD index 0x1801, subindex 2, see Table 2), and possibly stores it onboard so that it will be the default transmission-type after every subsequent reset or power-up.
The following transmission-types are supported:

- **PDO transmission type = 1**:
  after every socalled **SYNC** message issued on the CAN-bus the node sends 60 PDO messages, one message for every (configured) analog input channel.
  If the PDO's *inhibit time* is > 0 (OD index 0x1801, subindex 3) the PDOs containing locally stored conversion values will be sent in quick succession. If the PDO's *inhibit time* is equal to zero, a conversion has to be done for every channel so it can take up to several seconds before all PDOs have been sent (the ADC conversion rate can be as low as 3 Hz).
  The SYNC message is a CAN-message with a fixed COB-ID and no data bytes:

  Host → all (SYNC-)slave nodes

COB-ID
0x080

  Note that all nodes configured to respond to a SYNC will react to a SYNC message.

- **PDO transmission type = 253**:
  after every socalled Remote Transmission Request (**RTR**) for the PDO the node sends 60 PDO messages, one message for every (configured) analog input channel. Concerning the PDO's *inhibit time* the same applies as for transmission-type 1 (see above).
  The CAN *Remote Frame* that constitutes the RTR has no data bytes and looks like this:

  Host → SPICAN (node #5)

COB-ID
0x285 (0x280+*Node_ID*)

Note that an RTR is sent to and received by only one particular node.

- **PDO transmission type = 254**:
  same as transmission-type 253, except when the PDO's *inhibit time* is > 0 (OD index 0x1801, subindex 3), because then the node autonomously scans its analog input channels (with a frequency -per ADC- determined by the *inhibit time*) and a PDO is sent after every completed conversion. So after the node has been set into *Operational* state it continuously sends PDO messages. This can be stopped by putting the node into *Pre-operational* state: send an NMT message (see above) with byte 0 = 128 ('Enter Pre-operational state').

For all supported transmission types, assuming the SPICAN controller module is continuously scanning its analog input channels (i.e. *inhibit time* > 0, OD index 0x1801, subindex 3), the ADC value retrieved is the locally stored value of the last conversion of that channel. With a conversion frequency of 10 Hz (per ADC) and 30 input channels (per ADC) the last conversion could be up to 3 seconds 'old'; how old exactly is not known. If this is undesirable, the user should set the *inhibit time* to zero; a conversion is then started only after a request has been received (either through a SYNC or an RTR message).

For completeness it should be mentioned that individual analog input channels can –at any time– also be read out by reading the corresponding Object Dictionary entries (OD index 0x6404, see Table 3), using the *CANopen* **SDO** mechanism. Whether a locally stored value is retrieved or an analog-to-digital conversion takes place, again depends on whether the node is continuously scanning its input channels, or not.

2.3 Over-Limit Notification

Upper limits are implemented for the first 64 analog input channels (firmware versions 4.0 and up). How to set the limits is described in the next section. When a channel's ADC count exceeds its upper limit a PDO message is generated by the controller; when the count goes again below the upper limit another PDO message is generated. (*To be implemented(?)*: However, when the upper limit is exceeded it is temporarily decreased by 16 ADC-counts to add some hysteresis in order to prevent an unstable 'on/off' condition).

The PDO message contains the upper-limit state of 64 channels. It is possible that one PDO notification message contains a state change for more than 1 channel, so all 64 bits in the message have to be checked by the receiver of the PDO.

The upper-limit interrupt mechanism is active when the analog channels are scanned periodically by the controller (to be set in OD index 0x1801, subindex *3*: *2nd-transmit-PDO inhibit time)* and the interrupt is enabled (in OD index 0x6423).

A special version of the SPICAN application firmware has been made for monitoring Low-Voltage powersupplies in the **HERMES** experiment. In this system there are 24 ADCs with 8 multiplexed analog inputs each, monitoring voltage, current and temperature; in the over-limit check of this application only one of the inputs of each ADC is considered (this is a temperature-sensor input). If an over-limit situation is detected one of the CS5525 ADC's output signals is set, which in this particular system causes the powersupply to be switched off.

So in this application, although there are a total of 192 input channels, there are only 24 channels that have an upper-limit and over-limit check.

The *1st-transmit-PDO* message generated (COB-ID = 0x180 + Node_ID) contains 8 data bytes with the following syntax:

SPICAN (node #5) → Host

COB-ID	Byte 0	1	2	3	4	5	6	7
0x185	ch1-8 limit status	ch9-16 limit status	ch17-24 limit status	ch25-32 limit status	ch33-40 limit status	ch41-48 limit status	ch49-56 limit status	ch57-64 limit status

with a bit set to '1'signifying the upper limit has been exceeded by the corresponding channel. Within a byte, channels are mapped as follows (1 bit per channel):

Bit 7	6	5	4	3	2	1	0
ch #n+7	ch #n+6	ch #n+5	ch #n+4	ch #n+3	ch #n+2	ch #n+1	ch #n

2.4 Setting Upper Limits

At power-on or reset the upper limits for the analog channels are read from the onboard EEPROM if valid data is found there, otherwise the limit is set to the maximum ADC value (32767).

The analog inputs upper limits can be read and written by accessing OD index 0x6424 using the *CANopen **SDO*** mechanism.

Note that the interrupt message feature (using the *1st-transmit-PDO*) is currently only implemented for channels 1 to 64! (Enabling this for more channels requires the addition/implementation in the controller firmware of one PDO for every additional block of 64 channels).

Reading the upper limit of channel #3 requires the host to send the following message (OD index in byte 1+2, subindex in byte 3):

Host → SPICAN (node #5)

COB-ID	Byte 0	1	2	3	4	5	6-7
0x605	0x40	0x24	0x64	0x03	–	–	–

Assuming the upper limit is equal to 0x1234, the controller will reply with:

SPICAN (node #5) → Host

COB-ID	Byte 0	1	2	3	4	5	6-7
0x585	0x4B	0x24	0x64	0x03	0x34	0x12	–

Setting the upper limit of channel #3 to 0xABCD requires the host to send the following message:

Host → SPICAN (node #5)

COB-ID	Byte						
	0	1	2	3	4	5	6-7
0x605	0x2B	0x24	0x64	0x03	0xCD	0xAB	–

The controller will reply with:

SPICAN (node #5) → Host

COB-ID	Byte						
	0	1	2	3	4	5	6-7
0x585	0x60	0x24	0x64	0x03	–	–	–

The upper limit of all channels can be set to the same value by writing the limit value to channel number 0xFF (SDO message databyte 3).

In general an upper limit is read like this:

Host → SPICAN

COB-ID	Byte					
	0	1-2	3	4	5	6-7
0x600 + *Node_ID*	0x40	Object Index (0x6424)	Object Subindex (channel)	–	–	–

SPICAN → Host

0x580 + *Node_ID*	0x4B	Object Index (0x6424)	Object Subindex (channel)	ADC value (LSB first)	–

In general an upper limit is set like this:

Host → SPICAN

COB-ID	Byte					
	0	1-2	3	4	5	6-7
0x600 + *Node_ID*	0x2B	Object Index (0x6424)	Object Subindex (channel)	ADC value (LSB first)		–

SPICAN → Host

0x580 + *Node_ID*	0x60	Object Index (0x6424)	Object Subindex (channel)	–	–	–

2.5 Storing Parameters

Parameters and settings can be stored permanently onboard (in an EEPROM) by writing the string "save" to OD index 0x1010. Again the *CANopen* **SDO** mechanism is used to do this:

Host → SPICAN (node #5)

COB-ID	Byte							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0x605	0x23	0x10	0x10	*subindex*	0x73 ('s')	0x61 ('a')	0x76 ('v')	0x65 ('e')

with OD index 0x1010 in byte 1+2 and *subindex* in byte 3 with *subindex*:
=1: store all parameters (as listed for *subindex* 2, 3 and 4).
=2: store communication parameters, i.e. OD index 0x1801 (subindex 2,3).
=3: store analog channels upper limits, i.e. OD index 0x6424 (subindex 1 to 192).
=4: store ADC configurations, i.e. OD indices 0x2A00 to 0x2A18 (subindex 1 to 4), 0x2B00 to 0x2B18 (subindices 2 to 5), 0x2F00, 0x2F10 and 0x6423.

If the store-operation succeeded the controller sends the following reply:

SPICAN (node #5) → Host

COB-ID	Byte						
	0	**1**	**2**	**3**	**4**	**5**	**6-7**
0x585	0x60	0x10	0x10	*subindex*	–	–	–

If the store-operation did NOT succeed the controller sends the following reply (SDO *Abort Domain Transfer*, error reason: 'hardware fault' (for details see [6])):

SPICAN (node #5) → Host

COB-ID	Byte							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0x585	0x80	0x10	0x10	*subindex*	0	0	6 (Error Code)	6 (Error Class)

Parameters can be reset to their default values (by invalidating the corresponding contents of the EEPROM) by writing to OD index 0x1011, using this time the string "load" (0x6C, 0x6F, 0x61, 0x64) in bytes 4 to 7 of the **SDO**. Note that the default values take effect only after a subsequent reset of the node. Default values are listed in the OD tables in section 3.

2.6 ADC Reset and Calibration

At every power-on the controller (including its CAN-interface) and connected ADCs are reset, configuration parameters are read from EEPROM (if valid) and a calibration sequence is performed on all connected ADCs.

The following NMT message with command *Reset_Node* causes a full reset of the node including a reset and calibration sequence for all connected ADCs:

Host (NMT-Master) → SPICAN (NMT-Slave)

COB-ID	Byte 0	Byte 1
0x000	129 (*Reset_Node*)	5 (*Node-ID*) or 0 (all nodes)

In addition, it is possible to perform a reset and calibration sequence on an individual ADC by writing the number of the ADC to be reset to OD index 0x2C00 (using the SDO mechanism). See OD Table 5.

2.7 ADC Channel Numbering Scheme

At maximum 192 analog input channels are supported. These can be divided over a number of ADCs. The range of channel numbers reserved for one ADC is set in OD index 0x2F10 (this number has to be one of 8, 16 or 32, for reasons of computational convenience...). E.g. if set to 16 ADC#0 carries channel numbers 1 to 16, ADC#1 channels 17 to 32, etc etc., and ADC#12 channels 177 to 192. If set to 8, ADC#0 carries channels 1 to 8, ADC#1 channels 9 to 16, etc etc., and ADC#23 channels 185 to 192.

In OD index 0x2F00 can be set, up to which ADC number is actually in use; it is not necessary to set this parameter because the node detects which ADC numbers are connected and which are not (and skips these when initiating conversions). Setting this parameter properly, can speed up some of the node's actions.

The number of channels actually in use for each individual ADC can be set in the ADC-configuration object in OD index 0x2A*nn*, subindex 1; when the SPICAN controller scans the analog inputs, conversions are only performed for the set number of inputs.

Configuring the settings for a particular system is typically done just once (probably offline), after which the settings are stored onboard. This enables flexible interfacing to systems with different and variable numbers of CS5525-ADC based modules.

(**NB**: if settings cannot be stored because no EEPROM is present a customized firmware version can be made with default settings to match the application).

2.8 Emergency Objects

Emergency messages are triggered by the occurrence of a SPICAN internal (fatal) error situation. An emergency message has the following general syntax:

SPICAN → Host

COB-ID	Byte 0-1	Byte 2	Byte 3-7
0x080 + *Node_ID*	Emergency Error Code	Error Register (Object 0x1001)	Manufacturer specific error field

The following Emergency messages are defined for SPICAN:

Error Description	Emergency Error Code (byte 0-1)	Error Register (Object 1001H) (byte 2)	Manufacturer-specific Error Field (byte 3-7)
Watchdog or manual (front-panel) reset	0x6000	0x01	Byte 3,4,5,6: Manufacturer Device Name (Object Dictionary index 0x1008) Byte 7: 0
CAN-controller overrun: mes-sage lost	0x8100	0x10	Byte 3: 1 Byte 4: counter (modulo 256) Byte 5: CANSTA (CAN-controller status register) Byte 6,7: 0
CAN-controller error: communication error	0x8100	0x10	Byte 3: 2 Byte 4: counter (modulo 256) Byte 5: CANSTA (CAN-controller status register) Byte 6,7: 0
Local CAN message buffer overflow: message lost	0x8100	0x10	Byte 3: 3 Byte 4: counter (modulo 256) Byte 5: CANSTA (CAN-controller status register) Byte 6,7: 0
EEPROM: write failed	0x5000	0x80	Byte 3: 1 Byte 4,5,6,7: 0
EEPROM: read CRC error	0x5000	0x80	Byte 3: 2 Byte 4: parameter block for which CRC failed (2,3,4: according to OD 0x1010, subindex 2, 3 or 4) Byte 5,6,7: 0
ADC: conversion timeout	0xFF00	0x80	Byte 3: 1 Byte 4: ADC number (0..23) Byte 5,6,7: 0
ADC: reset failed	0xFF00	0x80	Byte 3: 2 Byte 4: ADC number (0..23) Byte 5,6,7: 0
ADC: offset calibra-tion failed	0xFF00	0x80	Byte 3: 3 Byte 4: ADC number (0..23) Byte 5,6,7: 0
ADC: gain calibration failed	0xFF00	0x80	Byte 3: 4 Byte 4: ADC number (0..23) Byte 5,6,7: 0

3 Object Dictionary

Table 1 to **Table 5** shows in detail the *CANopen* Object Dictionary (**OD**) of the Analog-Input SPICAN *CANopen* CAN-node with firmware **version 4.0** and later. The OD is based on the *CANopen* Device Profile for I/O modules [4], with device-specific OD entries to cover additional and specific features of the SPICAN system.

Column '**Attr**' shows the access rights attribute of an object: RO=read-only, RW=read-or-write, WO=write-only.

All entries in the SPICAN OD are accessed using the *CANopen* SDO mechanism with *expedited transfer* (object data content always <= 4 bytes).

Firmware **version 3.1** has the following limitations:

- OD entries related to the analog-in limit interrupt (OD entries 0x1800, 0x1A00, 0x6421 to 0x6424) are *not* supported
- OD entries related to parameter storage (OD entries 0x1010 and 0x1011) are *not* supported

The **CRYSTAL-CAN** *CANopen* hard- and firmware (any version) does *not* support Object Dictionary entry 0x1010 (*store parameters*) and 0x1011 (*restore default parameters*) due to the absence of an EEPROM.

Communication Profile Area (SPICAN)						
Index (hex)	**Sub Index**	**Name**	**Data/ Object**	**Attr**	**Default (hex)**	**Comment**
1000	-	Device type	U32	RO	00040191	Meaning: DSP-401 device profile, analogue inputs on device
1001	-	Error register	U8	RO	0	Error bits according to DS-301 (error status overview)
1002	-	Manufacturer status reg *	U32	RO	0	Error/time-out status of 8 ADCs
1004		#PDOs supported	Array			
	0	Total #PDOs supported	U32	RO	00000002	0 receive, 2 transmit PDO
	1	#PDOs sync	U32	RO	00000001	PDO after SYNC
	2	#PDOs async	U32	RO	00000002	PDO after RTR or 'event'
1008	-	Manufacturer device name	VisStr	RO	"SPIC"	= SPICAN module
100A	-	Manufacturer software version	VisStr	RO	"SC30"	SPICAN Version 3.0
100B	-	Node identifier	U32	RO		set by frontpanel hex-switches
100E	-	Node Guarding COB-ID	U32	RO	0x700+ Node-ID	According to CANopen Predefined Connection Set
100F	-	#SDOs supported	U32	RO	00000001	0 client, 1 server SDO
1010		Store parameters	Array			Save stuff in onboard EEPROM
	0	Highest index supported	U8	RO	4	
	1	Save all parameters	U32	RW	1	read: 1 write "save": store all
	2	Save communication parameters	U32	RW	1	read: 1 write "save": store PDO par's
	3	Save application parameters	U32	RW	1	read: 1 write "save": store analog limits
	4	Save application parameters	U32	RW	1	read: 1 write "save": store ADC configs
1011		Restore default parameters	Array			Invalidate stuff in onboard EEPROM
	0	Highest index supported	U8	RO	4	
	1	Restore all parameters	U32	RW	1	read: 1 write "load": invalidate all stored
	2	Restore communication parameters	U32	RW	1	read: 1; write "load": invalidate stored PDO par's
	3	Restore application parameters	U32	RW	1	read: 1; write "load": invalidate stored analog limits
	4	Restore application parameters	U32	RW	1	read: 1; write "load": invalidate stored ADC configs

Table 1. SPICAN Communication Profile Area of the CANopen Object Dictionary.

* See text for the layout of the Manufacturer Status Register.

Communication Profile Area (SPICAN) *(continued…)*						
Index (hex)	**Sub Index**	**Name**	**Data/ Object**	**Attr**	**Default (hex)**	**Comment**
1800		1st Transmit PDO parameters	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	2	
	1	COB-ID used by PDO	U32	RO	0x180+ Node-ID	According to CANopen Predefined Connection Set
	2	Transmission type	U8	RO	FE	254 decimal
1801		2nd Transmit PDO parameters	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	3	
	1	COB-ID used by PDO	U32	RO	0x280+ Node-ID	According to CANopen Predefined Connection Set
	2	Transmission type ♣	U8	RW	FD	253 decimal
	3	Inhibit time ♣ (in units of 100 µs)	U16	RW	0x3E8	If >0 node scans inputs with corresponding frequency (per ADC) Limitation: 0.2 Hz <= frequency <= 25 Hz (50000 >= inhibit time >= 400)
1A00		1st Transmit PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	
	1	Interrupt source: channel 1-32 bitmask	U32	RO	64220120	OD-index 6422, sub-index 1: Interrupt_Source_Bank_1 (see DSP-401); Size =32 bits
	2	Interrupt source: channel 33-64 bitmask	U32	RO	64220220	OD-index 6422, sub-index 2: Interrupt_Source_Bank_2 (see DSP-401); Size =32 bits
1A01		2nd Transmit PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	
	1	Multiplexor 1	U32	RO	6F100108	OD-index 6F10, sub-index 1: Multiplexor 1 (see DSP-404); Size = 8 bits
	2	24-bit analogue input	U32	RO	6404FD18	OD-index 6404, sub-index 253: Analogue input, via multiplexor; Size = 24 bits

Table 2. SPICAN Communication Profile Area of the CANopen Object Dictionary (continued).

♣ if *inhibit time* = 0: transmission type 254, 253 => conversion + PDO2 transmission of all channels after an RTR
transmission type 1 => conversion + PDO2 transmission of all channels after a SYNC
if *inhibit time* > 0: transmission type 254 => scan ADC(s), a PDO2 transmission after every conversion
transmission type 253 => scan ADC(s), PDO2 transmission of all channels after an RTR
transmission type 1 => scan ADC(s), PDO2 transmission of all channels after a SYNC

The Manufacturer Status Register (Object Dictionary index 0x1002), a 32-bit object, providing 4 bits per ADC of status information per ADC. The layout of this Register is as follows:

Bits	31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
ADC	#7	#6	#5	#4	#3	#2	#1	#0

The individual ADC status bits have the following meaning:

Bit 3	**Bit 2**	**Bit 1**	**Bit 0**
(not used)	Calibration error: - error during calibration procedure	Conversion error: - timeout waiting for conversion-ready	Reset error: - reset bit not set and/or - error in default register contents

Statuses for other ADCs (if more than 8 are present) can be found in OD index 0x2E00.

An ADC status of 0xf (all 4 bits are 1's) denotes that the ADC is not present in the configuration setting (OD index 0x2F00 *'Number of ADCs'*).

Standardised Device Profile Area (SPICAN)						
Index (hex)	**Sub Index**	**Name**	**Data/ Object**	**Attr**	**Default (hex)**	**Comment**
6404		Read analogue input Manufacturer-specific	Record			Here: 8 bits status, 16 bits analogue value
	0	Number of entries	U8	RO	192	Fixed, but actual hardware configuration may vary
	1	Input 1	I24	RO		1st analog input (24-bit)
	2	Input 2	I24	RO		2nd " " "
	…	…	…	…		…
	192	Input 192	I24	RO		192th " " "
	252	Multiplexor number	U8	RO	1	Defines which *mux* in the OD is used (DSP-404); but in this profile we don't define the *mux* itself
	253	Input via multiplexor	I24	RO		Read input #<mux1> (DSP-404)
6421		Interrupt Trigger Selection	Array			Reference to ways in which interrupts may be triggered: here available for completeness only
	0	Number of analog inputs	U8	RO	192	Upper limit interrupt supported on all input channels in principle
	1	Input 1	U8	RO	1	Bit 0: upper limit exceeded
	…	…	…	…	…	" " "
	64	Input 64	U8	RO	1	" " "
	65	Input 65	U8	RO	0	" " "
	…	…	…	…	…	" " "
	192	Input 192	U8	RO	0	" " "
6422		Interrupt source	Array			Determines which channel has produced interrupts: which channel has exceeded upper limit
	0	Number of bit banks	U8	RO	6	Space for 192 input channels
	1	Interrupt Source Bank 1	U32	RO		Bitmask for chan 1-32
	…	…	…	…		…
	6	Interrupt Source Bank 6	U32	RO		Bitmask for chan 160-192
6423	-	Global Interrupt Enable	Bool	RW	TRUE	Interrupt => PDO1 transmission
6424		Input Interrupt Upper Limit	Array			
	0	Number of analogue inputs	U8	RO	192	
	1	Upper limit input 1	U32	RW	32767	(Default = max. ADC value => no interrupt generated)
	…	…	…	…	…	…
	192	Upper limit input 192	U32	RW	32767	" " "
	255	Set upper limit all channels	U32	WO		Convenient entry for setting one limit for all channels

Table 3. Standardised Device Profile Area of the *CANopen* Object Dictionary for a device with CS5525 ADCs (providing 192 input channels, sufficient for six 32-channel ADCs, twelve 16-channel ADCs or twenty-four 8-channel ADCs).

Manufacturer-specific Profile Area (SPICAN)						
Index (hex)	**Sub Index**	**Name**	**Data/ Object**	**Attr**	**Default (hex)**	**Comment**
2A00		ADC-configuration [1] ADC#0	Record			
	0	Number of entries	U8	RO	8	
	1	Number of input channels	U8	RW	32	[0,32] and <= OD 0x2F10
	2	Conversion Word Rate	U8	RW	0	3-bit code [2]
	3	Input Voltage Range	U8	RW	0	3-bit code [3]
	4	Unipolar/Bipolar Measurement Mode	U8	RW	0	0 = bipolar, 1 = unipolar
	5	Power Save Mode	U8	WO		1 = power save
	6	Offset Register	U32	RW		CS5525 Offset Register
	7	Gain Register	U32	RW		CS5525 Gain Register
	8	D3-D0 pins	U8	RW		part of CS5525 Config Register
2A01		ADC-configuration ADC#1	Record			
…		…	…			
2A17		ADC-configuration ADC#23	Record			Max. 24 ADCs allowed
2B00		ADC-calibration-configuration ADC#0	Record			
	0	Number of entries	U8	RO	7	
	1	Conversion word rate during calibration	U8	RO	0	3-bit code [2] (always set to 15.02 Hz)
	2	Offset calibration type	U8	RW	5	3-bit code [4]
	3	Offset calib input channel	U8	RW	31	[0,31] and < OD-index 0x2F10
	4	Gain calibration type	U8	RW	6	3-bit code [4]
	5	Gain calib input channel	U8	RW	30	[0,31] and < OD-index 0x2F10
	6	Offset value	U32	RO		24-bits significant
	7	Gain value	U32	RO		24-bits significant
2B01		ADC-calibration-configuration ADC#1	Record			
…		…	…			
2B17		ADC-calibration-configuration ADC#23	Record			Max. 24 ADCs allowed

Table 4. Manufacturer-specific Profile Area of the *CANopen* Object Dictionary for a device with CS5525-ADCs.

[1] write access allowed **only** when ADC-input scanning not active (2nd-transmit-PDO *inhibit time* = 0)

[2] **000**: 15.02 Hz, **001**: 30.06 Hz, **010**: 60.01 Hz, **011**: 123.18 Hz,
100: 168.9 Hz, **101**: 202.27 Hz, **110**: 3.76 Hz, **111**: 7.51 Hz

[3] **000**: 100 mV, **001**: 55 mV, **010**: 25 mV, **011**: 1 V, **100**: 5 V

[4] **001**: offset self-calibration, **010**: gain self-calibration,
101: offset system-calibration, **110**: gain system-calibration

Manufacturer-specific Profile Area (SPICAN) ***(continued…)***						
Index (hex)	**Sub Index**	**Name**	**Data/ Object**	**Attr**	**Default (hex)**	**Comment**
2C00	-	ADC-reset-and-calibrate[5]	U8	WO	*n*	Reset ADC#*n* (0<=n<=23) and perform a calibration sequence
2D00	-	ADC-reset[5]	U8	WO	*n*	Reset ADC#*n* (0<=n<=23)
2E00		ADC status	Array			
	0	Number of status words	U8	RO	3	Space for 24 ADCs (4 bits/ADC)
	1	status ADC #0-#7	U32	RO		(= Object 0x1002) ADC error/time-out, etc.
	2	status ADC #8-#15	U32	RO		ADC error/time-out, etc.
	3	status ADC #16-#23	U32	RO		" "
2F00	-	Number of ADCs	U8	RW	6	To be set to highest number of ADC connected **NB:** (OD-index 2F00) * (OD-index 2F10) <=192
2F10	-	Reserved number of channels per ADC	U8	RW	32	One number for all ADCs; determines channel numbering scheme; can be set to 8, 16 or 32 only; actual number of channels in use to be set for each ADC individually in entry 0x2Axx, sub 1.

Table 5. Manufacturer-specific Profile Area of the *CANopen* Object Dictionary for a device with CS5525-ADCs (continued).

[5] write access allowed **only** when ADC-input scanning not active (2nd-transmit-PDO *inhibit time* = 0)

References

[1] **20CN592 80C592 based micro module with on board CAN bus controller,** User's Manual Rev1.3, Micro-key B.V., 1996.

[2] **CS5525/CS5526 16-bit / 20-bit multi-range ADC with 4-bit latch**, data sheet, Crystal Semiconductor Corporation, Sep 1996.

[3] CAN-in-Automation, **CANopen, CAL-based Communication Profile for Industrial Systems**, CiA DS-301, Version 3.0, Oct 1996.

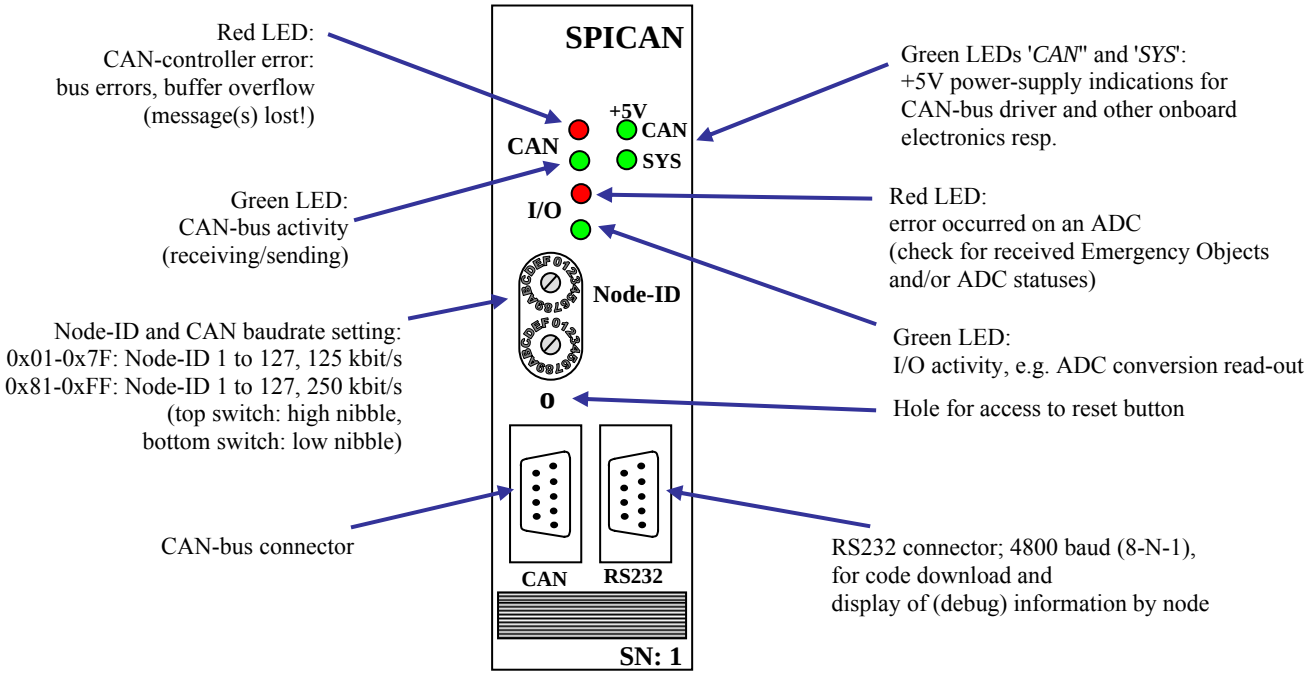
[4] CAN-in-Automation, **CANopen Device Profile for I/O Modules**, CiA DSP-401, Version 1.4, Dec 1996.

[5] CAN-in-Automation, **CANopen Device Profile for Measuring Devices and Closed-Loop Controllers**, CiA DSP-404, Revision 1.13, Nov 2 1998.

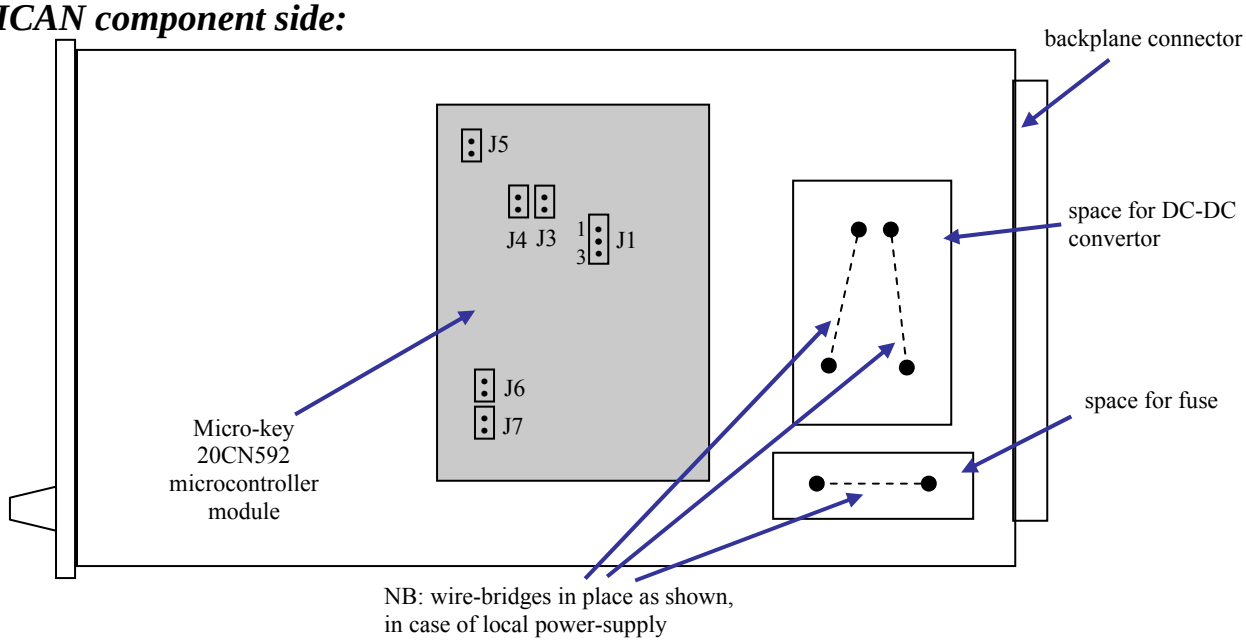
[6] H.Boterenbrood, **CANopen, high-level protocol for CAN-bus,** Version 2.0a, NIKHEF, Amsterdam, April 7 1999. (http://www.nikhef.nl/pub/departments/ct/po/doc/CANopen20.pdf).

APPENDIX A LEDs, Switches and Jumpers

SPICAN Frontpanel:

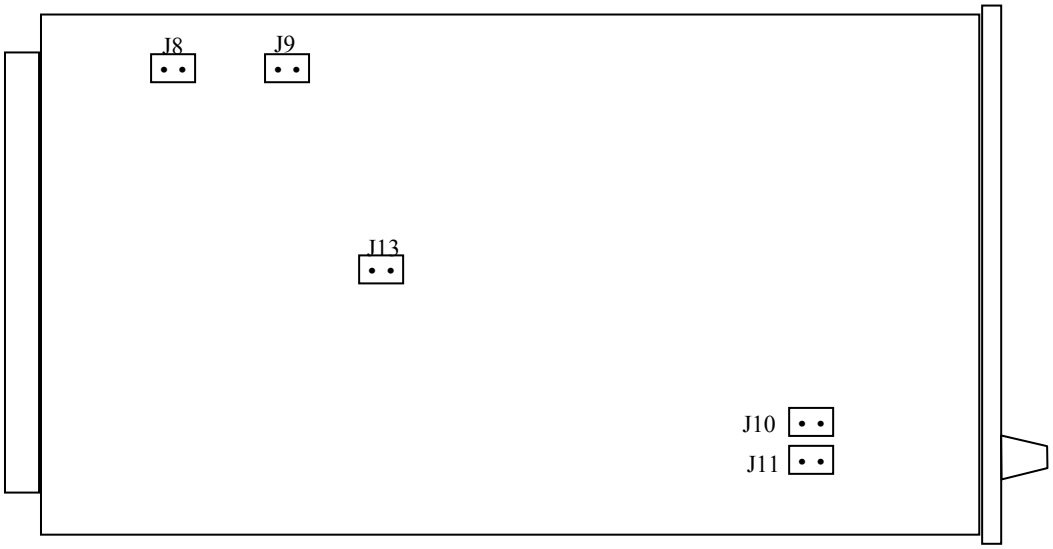


SPICAN component side:



J1: 80C592 internal watchdog (J1/1-2 closed: disabled; J1/2-3 closed: enabled).
J3: powerfail interrupt request via P1.0/#INT2 pin (open: interrupt disabled)
J4: external watchdog enable selector, 80C592 pin P1.1 to MAX691 (open watchdog disabled).
J5: reset jumper (closed: system will reset).
J6/J7: serial port signal connection/disconnection.

SPICAN solder side:



Power options:

Local power supply: via backplane
J8 + J9 closed, J10 + J11 open,
DC-DC convertor and fuse NOT placed, wire-bridges in place.

Ext. +5V power supply: via CAN-connector
J8 + J9 open, J10 + J11 closed,
DC-DC convertor NOT placed (wire-bridges in place), fuse placed.

Ext. +9…36V power-s.: via CAN-connector
J8 + J9 open, J10 + J11 closed,
DC-DC convertor and fuse placed.

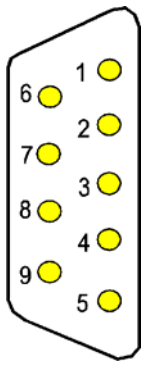
Additional power option:

Battery backup: J13 open.

APPENDIX B Connector Layout

9-pin D-sub *male* CAN-connector:

Pin	Signal
1	–
2	CAN_Low
3	V_gnd
4	–
5	–
6	–
7	CAN_High
8	–
9	V+



(backpanel connector layout to be provided)